Title	Deciding whether Graph G Has Page Number One is in NC
Author(s)	MASUYAMA, Shigeru; NAITO, Shozo
Citation	数理解析研究所講究録 (1992), 790: 155-161
Issue Date	1992-06
URL	http://hdl.handle.net/2433/82655
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

数理解析研究所講究録
第790巻 1992年 155-161

Deciding whether Graph G Has Page Number One is in NC

増山 繁 (豊橋技術科学大学知識情報工学系)
Shigeru MASUYAMA
Department of Knowledge-Based Information Engineering
Toyohashi University of Technology
Toyohashi 441, Japan

内藤 昭三 (NTT 基礎研究所)
Shozo NAITO
Basic Research Laboratories
NTT
Musashino 180, Japan

Abstract

Based on a forbidden subgraph characterization of a graph to have one page, we develop a polylog time algorithm to tell if page number of given graph G is one with polynomial number of processors, clarifying this problem to be in NC.

1 Introduction

This paper discusses the problem of deciding whether the given graph is noncrossing, where graph G is noncrossing if there exists a linear arrangement of vertices so that no pair of edges is crossing when they are drawn on the same side of the linear arrangement of vertices. Similar problem setting appears in the formulation of the noncrossing constraint on word modification to sentence generation in natural language processing, which motivates

us to study this problem.

This problem is a specialization of the book embedding[12] in the sense that this problem asks if the given graph has page number one, i.e., a graph can be embedded in a single page. In general, the book embedding is hard: it is NP-complete to tell if a planar graph can be embedded in two pages [3]. It is known[2, 3, 12] that non-crossing graphs on a single page are exactly the outerplanar graphs[6]. However, no polylog parallel decision algorithm to tell if a graph is outerplanar is known.

The problem of deciding whether given graph G has page number one is formally defined as follows:

Given graph $G = (V, E)$, decide whether G has page number one, where G has page number one if there exists a linear arrangement of vertices by which no pair of edges $e, e' \in E$ satisfies $s(e) < s(e') < t(e) < t(e')$, where $s(e)$ ($t(e)$, respectively,) is the smaller (the larger) end vertex of edge e in the arrangement.

We first illustrate a characterization of a graph to have page number one by two forbidden subgraphs. Based on this characterization we develop a polylog time algorithm with polynomial number of processors, clarifying this problem to be in NC.

Graphs considered in this paper are undirected and may have multiple edges. We also assume that a path denotes a simple path throughout this paper. CREW PRAM (see e.g., [5]) is adopted as a parallel computation model.

2 Forbidden Subgraph Characterization of a Graph to have Page Number One

We first introduce a characterization of outerplanar graphs[6].

Theorem 1.[6] Given graph $G = (V, E)$, G is outerplanar

if and only if

G has no subgraph homeomorphic to either K_4 or $K_{2,3}$, where K_4 is the complete graph on four vertices and $K_{2,3}$ is the graph illustrated in Fig. 1. □

Corollary 1. Given biconnected graph $G = (V, E)$, G is outerplanar

if and only if

G has no subgraph homeomorphic to either K_4 or $K_{2,3}$. □

As G has page number one if and only if G is outerplanar[2, 3, 12] (, we call this Fact 1), and K_4 is a forbidden subgraph of a series parallel graph[4] we have the following corollaries.

Corollary 2. If $G = (V, E, s, t)$ is a series parallel graph[4, 10] and has no subgraph homeomorphic to $K_{2,3}$, then G has a one page embedding in which the left terminal s of G has the smallest number. □

Corollary 3. Given series parallel graph $G = (V, E)$, G has page number one

if and only if

G has no subgraph homeomorphic to $K_{2,3}$. □

We now characterize a graph to have page number one for general graphs.

Combining Theorem 1 to Fact 1, we have the following Theorem.

Theorem 2. Given graph $G = (V, E)$, G is noncrossing

if and only if

G has no subgraph homeomorphic to either K_4 or $K_{2,3}$. □

No proof of Theorem 1 is provided in [6] and we supplement a constructive proof of Theorem 2, which ensures that Steps 2, 3 of the algorithm in the next section can be performed for each biconnected component B_i.

(Proof of Theorem 2) Necessity is obvious. To prove the sufficiency, we first note that the following three observations hold when G has no subgraph homeomorphic to K_4 or $K_{2,3}$.

Observation 1. Each biconnected component of G has no subgraph homeomorphic to K_4 and is a series parallel graph.

Observation 2. We can obtain a series parallel graph from each biconnected component of G by choosing any vertex as one of its terminal[7].

Observation 3. As each biconnected component of G has no subgraph homeomorphic to $K_{2,3}$, it has a one page embedding by which any vertex can be arranged first (by Observation 2 and Corollary 1 of Theorem 1).

Based on these observations, we shall prove the sufficiency by induction on

the number of biconnected components in G.

(1) If G has exactly one biconnected component, then the sufficiency holds by Corollary 1 of Theorem 1.

(2) We assume that the sufficiency holds when the number of biconnected components is $n-1$ and consider the case when the number of biconnected components of G is n.

Let $S(G)$ be a graph each of whose vertex corresponds to either a biconnected component of G or an articulate vertex of G, where (B, a) is an edge of $S(G)$ if and only if articulate vertex a belongs to biconnected component B. Note that $S(G)$ is a tree.

Let B be one of the leaves of $S(G)$(, note that B must correspond to a biconnected component of G,) and let a be an articulate vertex belonging to B. Let $G-B$ be a graph obtained by removing B, except a, from G. $G-B$ has $n-1$ biconnected components and has page number one. On the other hand, B has a one page embedding where a is arranged first. Thus by embedding this one page embedding of B in the one page embedding of $G-B$ between a and the vertex next to a in the arrangement, we have shown that G has page number one. □

3 Deciding if a Graph has Page Number One is in NC

We now introduce a linear time algorithm, by which we can decide, with respect to the number of edges, whether the given graph G has page number one. In the algorithm, $a(G)$ is the number of vertex disjoint paths with length at least two in G connecting its terminals and $b(G)=1$, if G has a subgraph homeomorphic
to G_5 in Fig. 2. Otherwise, $b(G)=0$.

Algorithm PNO

Step 1. Decompose G into biconnected components $B_1, B_2, \ldots, B_k$ and construct $S(G)$ defined in the proof of Theorem 2.

for each B_i, $i=1,\ldots,k$ let $G \leftarrow B_i$ and *do* Steps 2, 3:

Step 2. Decide whether G is a series parallel graph, and if G is a series parallel graph, then construct the parse tree of G which describes how to construct G from the set of edges of G[9, 10].

Step 3. Construct G in accordance with the parse tree in a bottom up manner.

Step 3.1. $a(e) \leftarrow 0$, $b(e) \leftarrow 0$, for each edge e of G.

Step 3.2. When series connection of G' and G'' is performed, let G be the resulting graph.

$$a(G) \leftarrow 1.$$

If either $a(G')=2$ or $a(G'')=2$

holds,

$$b(G) \leftarrow 1.$$

Step 3.3. When parallel connection is performed, let G be the resulting graph obtained from G' and G''. If either $b(G') = 1$ or $b(G'') = 1$ holds, then print "G has a page number more than one" and stop.

$$a(G) \leftarrow a(G') + a(G'')$$

If $a(G) \geq 3$, then print "G has page number more than one" and stop.

Step 3.4. If the root of the parse tree is already visited, then G has a page number one. □

The correctness of the algorithm immediately follows Corollary 1 of Theorem 1 and Theorem 2.

To implement the above algorithm in parallel, note that:

Step 1 can be performed in $O(log^2 n)$ time using $O(n^2/log^2 n)$ processors by [11], where n is the number of vertices in G.

"for statement" before Step 2 should be replaced by

for each B_i, $i = 1, \ldots, k$ *in parallel do* let $G \leftarrow B_i$ and *do* Steps 2, 3:

Step 2 can be performed $O(log^2\ n + log\ m)$ time with $O(n + m)$ processors, where n (m) is the number of vertices (edges) in G. by applying the recognition algorithm of series parallel graphs in [7] (note that although the first part of the algorithm in [7] may be simplified as G is biconnected, the overall complexity is not improved),

Step 3 can be computed in $O(log\ l)$ time with $O(l/log\ l)$ processors, where l is the number of vertices in the parse tree, by applying tree contraction algorithm in [1] (see also [7].) As l is $O(n)$, this step can be computed in $O(log\ n)$ time with $O(n/log\ n)$ processors,

In total, this algorithm works in $O(log^2 n + log\ m)$ time with $O(n^2/log^2 n + m)$ processors on CREW PRAM. □

4 Concluding Remarks

An actual one page embedding of given graph G which consists of more than one blocks can be obtained, if it exists, by appending the following Step 4 to Algorithm PNO.

Step 4. Merge one page embeddings of B_i's, in a manner described in the proof of Theorem 2, by visiting vertices of $S(G)$ in preorder from some arbitrary articulate vertex r.

Step 4 can be performed in parallel by obtaining preorder numbering of vertices of $S(G)$ and constructing lineary ordered list L of vertices of $S(G)$ where vertices are arranged in ascend-

ing order of its preorder numbering. This can be done by first applying Euler tour technique[11, 5] on tree $S(G)$ and doubling technique in a manner described in [5]. This can be done in $O(log\ k)$ time by $O(k)$ processors, where k is the number of biconnected components of G. As $k \leq n$, this step can be performed in $O(log\ n)$ time by $O(n)$ processors.

Then we replace, in parallel, each B_i by its one page embedding. To do this, note that each B_i can be replaced in the following manner as described in the proof of Theorem 2 :

for each articulate vertex j *in parallel do*

Make a one page embedding of each B_i beginning at j such that j is the father of B_i in the rooted tree obtained from $S(G)$ by Euler tour technique [5]. Then substitute, in parallel, each B_i with list L of vertices of the one page embedding. Finally, remove the articulate vertex j from the list L.

This can be done in constant time. Thus both the overall time complexity and the number of processors required coincide with those of Algorithm PNO.

References

[1] K. Abramson, N. Dadoun, D. G. Kirkpatrick and T. Przytycka, A simple parallel tree contraction algorithm, *J. of Algorithms 10, 287-302* (1989).

[2] F. Bernhart and P. C. Kainen, The book thickness of a graph, *J. Combin. Theory, Ser. B, 27,* pp. 320-331 (1979).

[3] F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg, Embedding graphs in books: A graph layout problem with applications to VLSI design, *SIAM J. Algebraic and Discrete Methods*, 1986.

[4] R. J. Duffin, Topology of series-parallel networks, *J. of Mathematical Analysis and Applications 10, 303-318* (1965).

[5] A. Gibbons and W. Rytter, *Efficient Parallel Algorithms*, Cambridge University Press (1988).

[6] F. Harary, *Graph Theory*, Addison-Wesley (1959).

[7] X. He, Efficient parallel algorithms for series parallel graphs, *J. of Algorithms 12, 409-430* (1991).

[8] X. He and Y. Yesha, Parallel recognition and decomposition of two terminal series parallel graphs, *Information and Computation 75, 15-38* (1987).

[9] J. E. Hopcroft and R. E. Tarjan, Dividing a graph into tri-

connencted components, *SIAM J. Comput., 2, 135-158* (1973).

[10] T. Kikuno, N. Yoshida and Y. Kakuda, A linear algorithm for the domination number of a series-parallel graph, *Discrete Applied Mathematics 5, 299-311* (1983).

[11] R. E. Tarjan and U. Vishkin, Finding biconnected components and computing tree functions in logarithmic parallel time, *SIAM J. Comput. 13*, 580-599 (1985).

[12] M. Yannakakis, Embedding Planar Graphs in Four Pages, *J. of Computer and System Sciences 38, 36-67*, 1989.

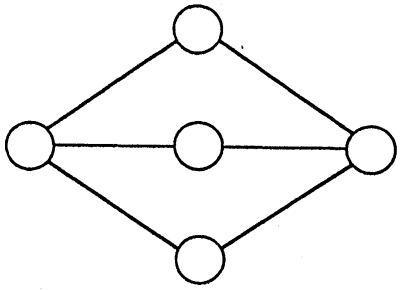

Fig. 1. Forbidden subgraph $K_{2,3}$

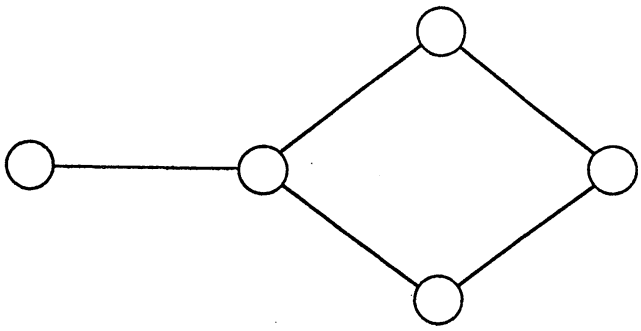

Fig. 2. G_5